REMARKS

Claims 1, 2 and 4-7 remain pending in the application.

Title of the Invention

The Examiner objects to the Title of the Invention as allegedly not clearly indicative of the invention to which the claims are directed.

The Applicants herein amend the title to be clearly indicative of the claims. The Applicants respectfully request that the objection to the Title of the Invention be withdrawn.

Claims 1, 2 and 4-7 over Kurowski

Claims 1, 2 and 4-7 were rejected under 35 U.S.C. §102(b) as allegedly being anticipated by U.S. Patent Application Pub. No. 2002/0019844 to Kurowski et al. ("Kurowski"). The Applicants respectfully traverse the rejection.

Claims 1, 2 and 4-7 recite, *inter alia*, determining with a service-chaining module of a first physical server an **identity** of a second physical server within a distributed environment **that stores** a requested application program.

The inventors appreciated that conventionally a request for a program and/or data from a first server that does not store the requested program and/or data fails. Failure of a request is very frustrating to a user of a client device. The user must then attempt to take further action to determine the location of a server that is able to service a request for a program and/or data. Many inexperienced users may not be able to determine a location of a server that is able to service their request, leaving them completely without solution. Applicants' claims overcome such deficiencies in the art. In accordance with the claimed features, a first server **determines** an identity of a second server that is **able** to service the request, eliminating the otherwise conventional frustration a user of a client device might otherwise experience.

Kurowski appears to teach a distributed computing system where large computational tasks are broken down into thousands of sub-tasks and

distributed to thousands of clients running on a variety of computers across the Internet. (see Abstract)

The Examiner alleges that Kurowski teaches determining with a service-chaining module of a first physical server an **identity** of a **second physical server** within a distributed environment **that stores** a requested application program at step 224 of Fig. 7.

Kurowski teaches:

> In step 220 (FIG. 7), the client 200 issues a request to the Task Server 1200 asking for a new task. This task may be further computation on one of the already existing application modules that the client has cached locally on the disk or it may require the download of a completely new application module. In either case, when the request is sent to the Task Server, the following information is preferably provided which assists the Task Server 1200 in assigning the most appropriate task to the client: Unique machine GUID; CPU number (optional: default is zero); User Id. Regarding the CPU number, if there are more than one processors then the client preferably needs to inform the Task Server which CPU needs the next task. This is because there might be different kinds of tasks running on different CPUs and this information would help the Task Server determine what is the next appropriate task assignment for the client. As mentioned above, application modules are also referred to herein as computation modules, computational modules, task modules, and/or simply modules. In return, in step 222 the Task Server 1200 assigns a task to the client 200 based on information received from the client 200. In step 224 the Task Server 1200 assembles module information relating to the assigned task and sends this module information to the client 200 in step 226. Specifically, the Task Server 1200 sends back the following information to the client 200 which the client needs to determine how it can run the next task: Unique Computational Module ID; Computational Module version number; URL to get the Computation Module if it is not already cached locally; and Checksum for computation module binary files to make sure they are still valid after being downloaded. (see Kurowski, paragraph [0154])
>
> In step 228 the client 200 uses the Computation Module URL to go to the File Server 1100, and in step 230 the client 200 downloads a self-extracting EXE file. This file is preferably downloaded in the WINDOWS.backslash.TEMP folder based on the environment of the computer. Then, the checksum obtained earlier from the Task Server 1200 is compared with the checksum of this self extracting EXE file. If it appears valid, then, the self extracting EXE is run and all its files are extracted in, for example, a WINDOWS.backslash.TEMP.backslash. ENTROPIA folder. Then, these module files which are most likely DLLs are copied to the appropriate module folder based on the ModuleID and its

> version number. In this way the client 200 downloads a Task Module (or computation or application module). (see Kurowski, paragraph [0156])

Thus, Kurowski teaches a Task Server that, in response to a client request for a new task, assigns a most appropriate task to a client based on client provided information. A URL is sent to the client to download the most appropriate task. Kurowski's Task Server **knows** where the most appropriate task is stored, i.e., at the File Server. **Knowing** where data is stored obviates the need to **determine** where data is stored, much less determine an **identity** of a **second physical server** within a distributed environment **that stores** a requested application program, as required by claims 1, 2 and 4-7.

Claims 1, 2 and 4-7 recite, *inter alia*, transmitting a message object **from** a first physical **server** **to** a second physical **server** to **enable** the second physical server to transmit an application program to a client device in response to the client device request transmitted to the first physical server.

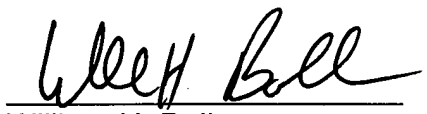
As discussed above, Kurowski's File Server transmits a most appropriate task to a client in response to a client device downloading the most appropriate task (through use of a URL transmitted to the client from the Task Server). Kurowski's File Server is simply a repository of tasks that needs no **enablement** to transmit information to a client. Kurowski fails to teach a message object that is transmitted from the Task Server to the File Server to **enable** the File Server to transmit the most appropriate task to the client. Kurowski fails to teach transmission of a message object between **servers** to **enable** a second server to transmit information to a client, much less transmitting a message object **from** a first physical **server** **to** a second physical **server** to **enable** the second physical server to transmit an application program to a client device in response to the client device request transmitted to the first physical server, as recited by claims 1, 2 and 4-7.

Accordingly, for at least all the above reasons, claims 1, 2 and 4-7 are patentable over the prior art of record. It is therefore respectfully requested that the rejection be withdrawn.

Conclusion

All objections and rejections having been addressed, it is respectfully submitted that the subject application is in condition for allowance and a Notice to that effect is earnestly solicited.

Respectfully submitted,



William H. Bollman
Reg. No.: 36,457
Tel. (202) 261-1020
Fax. (202) 887-0336

MANELLI DENISON & SELTER PLLC
2000 M Street, NW 7TH Floor
Washington, DC 20036-3307
TEL. (202) 261-1020
FAX. (202) 887-0336

WHB/df